RVCQ3D - Rotor Viscous Code Quasi-3-D

User's Manual and Documentation

Version 303, March 1, 1999

Dr. Rodrick V. Chima
NASA Glenn Research Center, MS 5-10
21000 Brookpark Road
Cleveland, Ohio 44135 USA

phone: 216-433-5919
fax: 216-433-5802
email: fsrod@grc.nasa.gov
internet: http://www.grc.nasa.gov/WWW/5810/webpage/rvc.htm

Abstract

RVCQ3D (Rotor Viscous Code Quasi-3-D) is a rapid computer code for analysis of inviscid or viscous blade-to-blade flows in turbomachinery. It accounts for the quasi-three-dimensional effects of rotation, radius change, and stream surface thickness variation. The code has been heavily used at NASA Glenn and in U.S. industries and universities for a wide variety of problems including analysis of turbopump blades for the Space Shuttle main engine, analysis of roughness effects on fan blades, and the design of supersonic fan blades. The code runs quickly on most computers. This report serves as the user's manual and documentation for the RVCQ3D code. The code and some aspects of the numerical method are described. Steps for code installation and execution are given. The grid, input, and output variables are described in detail.

Introduction

RVCQ3D (Rotor Viscous Code Quasi-3-D) is a rapid computer code for analysis of quasi-three-dimensional viscous flows in turbomachinery. RVCQ3D is applicable to most types of turbomachinery including compressors and turbines with axial, radial, or mixed flows. Several example solutions are shown in figures 1–4. The code solves the thin-layer Navier-Stokes equations on a blade-to-blade surface of revolution using an explicit finite-difference technique. Three turbulence models are available. Several schemes are used to accelerate convergence. The code is written in Fortran and runs quickly on most computers. Solution files are compatible with most CFD flow visualization packages.

Reference 1 describes the mathematical formulation of the problem, and an explicit multigrid solution scheme that was used in previous versions of the code. The current release of the code uses an explicit multistage Runge-Kutta solution scheme based on the work of Jameson, Schmidt, and Turkel. A spatially-varying time step and implicit residual smoothing are used to accelerate convergence. Preconditioning is used to accelerate convergence for low speed flows.

The problem is formulated by writing the Euler or Navier-Stokes equations on a surface of revolution in an (m,θ) coordinate system (fig. 5.) The m-coordinate is the arc length along the surface, and the θ-coordinate is in the circumferential direction. The stream surface radius and thickness are assumed to be known functions of m, usually obtained from an axisymmetric throughflow analysis. Terms are included for rotation around the m-axis.

The equations are mapped to a body-fitted (ε, η) coordinate system using standard finite-difference techniques. All viscous terms are dropped in the streamwise (x) direction using the thin shear layer approximation. Turbulence effects are modeled using the Baldwin-Lomax, Cebeci-Smith, or Wilcox k-ω turbulence models. The equations are differenced using second-order finite differences throughout.

.

n	α^1	α^2	α^3	α^4	α^5	λ^*
2	1.2	1.				.95
3	.6	.6	1.			1.5
4	.25	.3333	.5	1.		2.8
5	.25	.1667	.375	.5	1.	3.6

Table 1 - Runge-Kutta parameters $\alpha^1 - \alpha^5$ and maximum Courant number λ^* for *n*-stage schemes.

C-type grids are used to give good resolution of blade leading-edges and wakes (see figure 6.) Grid input is in standard PLOT3D xyz-file format, so any C-grid generator can be used. However, the GRAPE code developed by Reece Sorenson at NASA Ames Research Center has been modified for turbomachinery by the author, and works directly with RVCQ3D.

An explicit multistage Runge-Kutta scheme is used to solve the difference equations by marching in time from an initial guess to a steady-state solution. The user may choose the number of stages in the scheme, but a four stage scheme is recommended. A spatially-varying time step is used to accelerate the convergence rate, but the code may be run with a constant time step and second-order time accuracy by changing an input flag. The explicit scheme has a stability limit on the time step (a Courant number of about 2.8 for a four stage scheme.) Implicit residual smoothing is used to increase the allowable time step and thus increase the convergence rate. Preconditioning may also be used to accelerate convergence for low-speed (incompressible) flows.

RVCQ3D is written completely in Fortran and runs as a batch job on most PC's, workstations, or mainframe computers. Namelist input data must be supplied to RVCQ3D as an ascii dataset. Typical viscous solutions take 1500-2000 iterations to converge and require a few minutes on a PC or workstation, depending on grid size and flow characteristics. Printed output consists of a residual history, spanwise profiles of blade-to-blade averaged quantities at the grid inlet and exit, and streamwise profiles of various quantities on the blade surfaces. No graphical output is provided, but the solution files can be read directly and plotted using the public domain CFD visualization codes PLOT3D and FAST**,** or the commercial codes ENSIGHT, FIELDVIE, and TECPOT.

This documentation briefly describes how the RVCQ3D code works. Instructions for dimensioning, compiling, and running RVCQ3D are given for Silicon Graphics (SGI) workstations and Cray mainframes. The namelist input variables are described in detail. Finally, the structure of the output file is described.

Four test cases are included: a subsonic turbine stator, a linear compressor cascade, a centrifugal impeller, and an annular compressor cascade (figs. 1–4.) These cases cover most of the input options available in the code.

Non-Dimensionalization

The grid *xyz*-file may be input in any arbitrary units of length. The input parameters to RVCQ3D and the output solution file are strictly nondimensional, with the exception of lengths which must be input in the same units as the grid.

All quantities are nondimensionalized by an arbitrary reference stagnation state defined by total density ρ_0 and total sonic velocity c_0. A reference viscosity μ_0 is defined by the stagnation temperature $T_0 = c_0^2/(\gamma R)$. Standard conditions are often used for the reference state, but *any* self-consistent state may be used as long as the units of length are consistent with the grid units. The reference state is **not necessarily** the inlet state.

For input and output, pressures and temperatures are nondimensionalized by P_0 and T_0. Within the code, pressures are usually nondimensionalized by $\rho_0 c_0^2 = \gamma P_0$.

Four input parameters must be properly nondimensionalized: *renr, om, prat,* and *tw* (See "RVCQ3D Input", pp. 7.)

Numerical Method

Multistage Runge-Kutta Scheme

Multistage schemes were developed by Jameson, Schmidt, and Turkel (11) as a simplification of classical Runge-Kutta integration schemes for ODE's. The simplification reduces the required storage, but also reduces the time-accuracy of the schemes, usually to second order. The following discussion of these schemes should give some guidance in choosing parameters for running the code.

The *k*th-stage of an *n*-stage scheme may be written as:

$$q^k = q^0 - \alpha^k \Delta t (R_I^k + R_V{}^0)$$

where q is an array of five conservation variables (See "Solution Q-File", pp. 14.), k is current the stage, q^0 is the previous time step, α^k are the multistage coefficients discussed below, Δt is the time step, R_I^k is the inviscid part of the residual, and $R_V{}^0$ is the viscous part of the residual including the artificial dissipation. Note that R_I^k is evaluated at every stage *k*, but $R_V{}^0$ is evaluated only at the initial stage for computational efficiency.

The maximum stable Courant number λ^* for an n-stage scheme can be shown to be $\lambda^* = n - 1$. The actual stability limit depends on the choice of α^k. For consistency α^n must equal 1, and for second-order time accuracy α^{n-1} must equal *1/2*. The values of α^k used in the code and the theoretical maximum Courant number λ^* are set by a **data** statement in subroutine setup and are given in table 1.

The number of stages is set with the variable *nstg*. Using $nstg = 4$ is recommended, although Jameson et. al. tend to favor 5 stages.

A spatially-varying Δt is used to accelerate the convergence of the code. Setting $ivdt = 1$ causes the Courant number to be set to a constant (input variable *cfl*) everywhere on the grid, and to be recalculated every *icrnt* iterations. This option is highly recommended. Set *icrnt* to a moderate number, e.g. 10, so that the time step is recalculated occasionally. The time step is recalculated when the code is restarted and may cause jumps in the residual if *icrnt* is too big.

Implicit residual smoothing (described later) may be used to increase the maximum Courant number by a factor of two to three, thereby increasing the convergence rate as well.

Artificial Viscosity

The code uses second-order central differences throughout and requires an artificial viscosity term to prevent odd-even decoupling. A fourth-difference artificial viscosity term is used for this purpose. This term is third-order accurate in space and thus does not affect the formal second-order accuracy of the scheme. The input variable *avisc4* scales the fourth-difference artificial viscosity, and should be set between 0.25 and 2. A good starting value is $avisc4 = 1.0$. If the solution is wiggly, increase *avisc4* by 0.5. If it is smooth, try reducing avisc4 by 0.5. Larger values of *avisc4* may improve convergence somewhat, but the magnitude of *avisc4* has little effect on predicted losses or efficiency.

The code also uses a second-difference artificial viscosity term for shock capturing. The term is multiplied by a second difference of the pressure that is designed to detect shocks. Note that the second-difference artificial viscosity is first-order in space, so that the solution reduces to first-order accurate near shocks. Two other switches developed by Jameson (11) are used to reduce overshoots around shocks. The input variable *avisc2* scales the second difference artificial viscosity, and should be set between 0. and 2. Use 0. for purely subsonic flows, and start with 1. for flows with shocks. If shocks are wiggly, increase *avisc2* by 0.5. If they are smeared out, try decreasing *avisc2* by 0.5. Shocks will be smeared over four or five cells regardless of the value of *avisc2*. The magnitude of *avisc2* also has little effect on predicted loss or efficiency.

Eigenvalue scaling described in (2) is used to scale the artificial viscosity terms in each grid direction. This greatly improves the robustness of the code. The artificial viscosity is also reduced linearly by grid index near walls to reduce its effect on the physical viscous terms. Input variable *jedge* is the index where the linear reduction begins.

For computational efficiency the artificial viscosity is usually calculated only at the first stage of the Runge-Kutta scheme. This works well for most problems, but for difficult problems the robustness of the scheme can be improved by updating the artificial viscosity more often. This is done by setting $ndis = 2$ with $nstg = 4$ or 5. Then for $nstg = 4$ the artificial viscosity is calculated at stages 1 and 2. For $nstg = 5$ the artificial viscosity is calculated at stages 1, 3, and 5. In RVCQ3D the physical viscous terms are calculated at the same time as the artificial viscosity. Thus, setting $ndis = 2$ increases the CPU time per stage significantly, but it is so reliable that it is usually the preferred scheme.

Implicit Residual Smoothing

Implicit residual smoothing was introduced by Lerat in France and popularized by Jameson in the U.S. as a means of increasing the stability limit and convergence rate of explicit schemes. The idea is simple: run the multistage scheme at a high, unstable Courant number, but maintain stability by smoothing the residual occasionally using an implicit filter. The scheme can be written as follows:

$$(1-\varepsilon_\xi \delta_{\xi\xi})(1-\varepsilon_\eta \delta_{\eta\eta})\bar{R} = R$$

where ε_ε and ε_η are constant smoothing coefficients in the body-fitted coordinate directions ε and η indicated in figure 6. Here δ is a second-difference operator, $\bar{R}$ is the smoothed residual, and R is the unsmoothed residual.

It can be shown that if the scheme converges implicit residual smoothing does not change the solution. Linear stability theory shows that the scheme can be made unconditionally stable if ε_i is big enough, but also shows that the effects of the artificial viscosity are diminished as the Courant number is increased. In practice the best strategy seems to be to double or triple the Courant number of the unsmoothed scheme. If the residual is smoothed after every stage, the theoretical 1-D values of ε_i needed for stability are given by:

$$\varepsilon_i \geq \frac{1}{4}\left[\left(\frac{\lambda}{\lambda^*}\right)^2 - 1\right]$$

where λ^* is the Courant limit of the unsmoothed scheme (given in table 1,) and λ is the larger operating Courant number. For example, to run a four-stage scheme at a Courant number $\lambda = 5.6$, the smoothing coefficient should be:

$$\varepsilon_i \geq \frac{1}{4}\left[\left(\frac{5.6}{2.8}\right)^2 - 1\right] = 0.75$$

A single variable $eps = \varepsilon$ is input to RVCQ3D. The 1-D limit given above usually gives a reasonable estimate for ε, but the code will converge best when ε is minimized. Values of ε_ε and ε_η are evaluated at each grid point within the code by scaling *eps* using the same Eigenvalue scaling coefficients used for the artificial dissipation. This has proven to be quite robust.

In older versions of RVCQ3D the residual smoothing coefficients were set as constants in each direction. The Eigenvalue scaling described above almost always works better, but constant smoothing coefficients can be set using input variables *epi* and *epj*.

Implicit residual smoothing involves a scalar tridiagonal inversion for each variable along each grid line in each direction. It adds about 20 percent to the cpu time when applied after each stage. Smoothing can be done after every other stage to reduce cpu time (about 7 percent,) but *eps* must be increased (approximately doubled.) This option is not recommended.

Recommended Numerical Parameters

Table 2 lists recommended parameters for three numerical schemes, in order of the author's preference. For each scheme $avisc2 = 1.0$, $avisc4 = 0.5$, and $irs = 1$.

Preconditioning

Density-based schemes like RVCQ3D solve the continuity equation by driving the density residual to zero. For low speed (nearly incompressible) flows the density residual is naturally near zero, and the schemes fail to converge. Preconditioning, described in references (8) and (9) improves the convergence rate in two ways. First, it replaces the q-variables $q = [\rho, \rho u, \rho v, e]$ with variables that are better-behaved at low speeds $W = [p, \rho u, \rho v, h]$, where p is the pressure and h is the total enthalpy. Second it multiplies the equations by a matrix designed to equalize the wave speeds of each equation. The preconditioning matrix has the local flow velocity in the denominator and must be limited when the velocity becomes small. The preconditioning operator is designed so that it has no effect on the steady-state solution.

Preconditioning works extremely well for the Euler equations and less well for the Navier-Stokes equations. It will allow solutions for very low Mach number flows to converge when they would not converge otherwise.

Turbulence Models

Three turbulence models are available in RVCQ3D, the Baldwin-Lomax and Cebeci-Smith algebraic models, and the Wilcox two-equation k-ω models. All three models include transition models and surface roughness effects.

nstg	*ndis*	*cfl*	*eps*	comments
4	2	5.6	0.75–1.00	good overall scheme
5	2	7.0	1.25–1.50	slow per stage, fast convergence
2	1	2.5	1.35–1.60	fast per stage, slow convergence

Table 2 - Recommended numerical parameters for three Runge–Kutta schemes. For each scheme $avisc2 = 1.0$, $avisc4 = 0.5$, and $irs = 1$.

Baldwin-Lomax and Cebeci-Smith Turbulence Models

The Baldwin-Lomax model is accessed by setting input variable $ilt = 2$. The model is implemented as described in the original reference (3), except that two constants have been changed to $C_{Cp} = 1.216$ and $C_{Kleb} = 0.646$. The length scale for the Baldwin-Lomax model is correlated to the maximum of a function $f = y|\omega|D$, where y is the distance from the wall, $|\omega|$ is the magnitude of the vorticity, and D is the Van Driest damping function. In some cases that maximum is not well-behaved, so RVCQ3D limits the search to the grid line at j-index *jedge* away from the body. Input variable *jedge* should be chosen slightly larger than the largest extent of the boundary layer. Solutions are usually insensitive to values of *jedge.*

The Cebeci-Smith model is accessed by setting input variable $ilt = 3$. The model is implemented like the Baldwin-Lomax model except that the length scale is found by integrating $\int_0^{\delta} f\,dy = \delta^* u_e$, as described in (4). Here the upper bound of the integral is usually found automatically since $f \to 0$ as $y \to \delta$. However, cases with free-stream vorticity can have a finite *f* outside the boundary layer, so input variable *jedge* is used to bound the integral in RVCQ3D. The Cebeci-Smith model is very reliable for turbine heat transfer problems, but is not recommended for transonic compressors which tend to produce free-stream vorticity behind the bow shock.

Transition is predicted in both models at the location where $\mu_{turb}/\mu_{lam} > cmutm$, where *cmutm* is an input variable usually set to 14, as recommended in (3). The model is crude, but often works surprisingly well for moderate Reynolds numbers and free-stream turbulence.

Roughness effects are included in both models using the Cebeci-Chang roughness model (5). The model modifies the turbulent length scale based on the equivalent sand-grain roughness height in turbulent wall units *hrough*. Input variable *hrough* can be estimated using $hrough \approx 10,000 \times h_{RMS}/chord$, where h_{RMS} is the RMS roughness height. If $hrough < 4.6$, the surface is hydraulically smooth and roughness effects are not modeled.

Wilcox k-ω Turbulence Model

The Wilcox k-ω is accessed by setting input variable $ilt = 4$ or 5. The model is described in (6), and implemented as described in (7) using a first-order upwind ADI scheme. Two versions of the model are included, a baseline model $(ilt = 4)$ and a low Reynolds number model $(ilt = 5)$. The baseline model gives a fully-turbulent solution, and, unlike k-ε models, it is valid all the way to the wall. The low Reynolds number model includes transition effects.

Three input parameters effect the k-ω model, the surface roughness *hrough* as described above, the free-stream turbulence level *tintens* (in percent), and the free-stream turbulence length scale *tlength*. Solutions are generally insensitive to *tlength* except for the location of transition.

Compiling and Running RVCQ3D

RVCQ3D is supplied as a unix script which generates the source and include files and compiles them. The format of the script file is shown below.

```
#! /bin/csh -f
cat > RVCQ3D.f << '/eof'
#RVCQ3D source code goes here
'/eof'
#------------------------------------------------------
cat > sec.f << '/eof'
#replacement for Cray-specific timing routine here
'/eof'
#------------------------------------------------------
cat > csca << /eof
#scalar common block goes here
eof
#------------------------------------------------------
#other common blocks follow
#------------------------------------------------------
cat > param << /eof
#code dimensioning parameters go here
      parameter (id=385,jd=64
/eof
#------------------------------------------------------

#compiler commands with options go here
/bin/rm...
```

On a unix platform, edit the script and go to the bottom. Change the parameter statements to values greater than or equal to the size of the grid to be run. (See *Parameter Statements* below.) Comment, uncomment, or add compilation commands appropriate for the computer to be used. (See below for compilation commands for SGI and Cray computers.) Save the script, set execute permission, and execute it.

On a PC, manually strip out, save, and compile the files between the *cat* and */eof* commands.

Parameter Statements

RVCQ3D uses a parameter statement to make redimensioning simple. The parameter statement and labeled common blocks are inserted during compilation using Fortran include statements. A typical parameter statement is shown above. Actual parameter values may be different in the distribution code. RVCQ3D checks the user input against the dimensioning parameters and stops with a fatal error message if the code is not dimensioned properly.

For a C-grid dimensioned (m, n), the dimension parameters must have $id \geq m$ and $jd \geq n + 1$. The j-direction requires an extra line of storage for a dummy grid line used in RVCQ3D to implement the periodic boundary conditions.

Thirty-six variables are stored at each grid point. Thus the computer memory required to run RVCQ3D is $36 \times id \times (jd + 1)$ words for arrays, plus about 200K words for the executable code.

Compiling RVCQ3D on Silicon Graphics Workstations (IRIX)

The following commands can be used to compile RVCQ3D on various SGI workstations:

```
#SGI R8000,R10000 CPU
f77 -O3 -mips4 -WK,-o=0,-so=2,-ro=0 -OPT:round=3:IEEE_arith=3 \
-lfastm -o RVCQ3D RVCQ3D.f noncray.f
strip RVCQ3D

#SGI R4000 CPU (Indigo 2)
f77 -O2 -sopt -mips2 -lfpe -o RVCQ3D RVCQ3D.f
strip RVCQ3D
```

Compiling RVCQ3D on Cray Research Computers (UNICOS)

The following command will compile RVCQ3D on a Cray C-90 with autotasking and full optimization:

```
f90 -O3 -o RVCQ3D RVCQ3D.f
```

Running RVCQ3D

The executable program is run as a standard unix process:

```
RVCQ3D < std_input > std_output &
```

Standard Input, Output, and Binary Files

An ascii input file for RVCQ3D is read from Fortran unit 5 (standard input.) Printed output from RVCQ3D is written to Fortran unit 6 (standard output.) Binary files linked to Fortran units 1-3 may be used in the execution of RVCQ3D, depending on input options. The units are used as follows:

fort.1input grid file, required. See “Grid XYZ-File”, pp. 14.
fort.2 input solution file, read if $iresti = 1$. See “Solution Q-File” pp. 14.
fort.3 output solution file, written if $resto = 1$. See “Solution Q-File” pp. 14.
fort.7 input k-ω turbulence file, read if $iresti = 1$ and $ilt \geq 4$.
fort.8 output k-ω turbulence file, written if $resto = 1$ and $ilt \geq 4$.

All files are treated as unformatted binary files. They are not explicitly *opened* in the code. Files are usually linked to file names before running RVCQ3D,

```
ln input.grid.file fort.1
ln input.solution.file fort.2
ln output.solution.file fort.3
etc.
```

Binary grid files can be used immediately on the same type of computer on which they were generated. Files generated on an SGI machine can be read into PLOT3D using the *read /unformatted* option. Files generated on a Cray can be converted to SGI format in one of two ways:

1. By using the *itrans* command at NASA Ames to convert the files to SGI binary. Files converted using *itrans* can be read into PLOT3D using the *read /binary* option <default>.

```
itrans file.xyz file.SGI.xyz
```

2. By *assigning* the files as 32 bit ieee binary files on the Cray before execution. The lower precision does not affect the accuracy of RVCQ3D. Files written on a Cray while assigned as ieee binary can be used directly on an SGI machine and can be read into PLOT3D using the *read /unformatted* option.

```
assign -F f77 -N ieee fort.1
assign -F f77 -N ieee fort.2
etc.
```

RVCQ3D Input

Defaults are given in angle brackets, <Default=value> or <default.> If no default is given, the value **MUST** be input.

Title

ititle An alphanumeric string of 80 characters or less printed to the output. The character string must be enclosed in single quotes. The following Fortran input statement is used to read *ititle*:

```
read *,ititle
```

***&nam1* - Grid Size Parameters**

m — Grid size in i- (streamwise) direction. Must agree with *m* read from the grid file. The code must be dimensioned with $id \geq m$.

n — Grid size in j- (blade-to-blade) direction. Must agree with *n* read from the grid file. The code must be dimensioned with $jd \geq n + 1$ to allow for a dummy grid line added internally to implement the periodic boundary conditions.

mtl — i-index of lower trailing-edge point on a C-grid. *Mtl* is printed at the end of the GRAPE output. *Mtl* can be 1 for a blade that extends to the exit of the grid.

mil — i-index of the last periodic point on the outer boundary. Also acts as the lower point on the inlet boundary. *Mil* is printed at the end of the GRAPE output.

ich — Cascade/flat plate analysis flag.

= 0 Cascade analysis on a C-grid <default.>

= 1 Flat plate analysis on an H-grid. Simple geometry only used for testing turbulence models.

***&nam2* - Algorithm Parameters**

nstg — Number of stages for the Runge-Kutta scheme, usually 4, but can be 2-5. <4>

ndis — Number of evaluations of the artificial and physical dissipation terms during the Runge-Kutta scheme. Generally 1, but larger values improve the robustness of the algorithm. See “Artificial Viscosity”, pp. 3. <1>

If *ndis* = 2 and *nstg* = 4 the dissipative terms are updated after stages 1 and 2.

If *ndis* = 2 and *nstg* = 5 the dissipative terms are updated after stages 1, 3, and 5.

avisc2 — Second-order artificial dissipation coefficient. Typically 0.0 - 2.0. Use 0.0 for purely subsonic flow or 1.0 for flows with shocks. <1.0.>

avisc4 — Fourth-order artificial dissipation coefficient. Typically 0.25 - 1.5. Start at 1.0 and reduce *avisc4* to 0.5 if possible. <0.5.>

irs — Implicit residual smoothing flag. Usually = 1. (See “Implicit Residual Smoothing”, pp. 4.)

= 0 No residual smoothing

= 1 Implicit smoothing after every Runge-Kutta stage <default.>

= 2 Implicit smoothing after every other stage. *eps* must be increased (roughly x 2) for this option to work.

eps — Overall implicit smoothing coefficient based on the 1-D stability limit (See “Implicit Residual Smoothing”, pp. 4.) RVCQ3D will calculate the 1-D limit if *eps* is defaulted.

epi, epj — Implicit smoothing coefficient multipliers for the *i*, and *j* directions. (See “Implicit Residual Smoothing”, pp. 4.) Rarely used. <1.>

ivdt — Variable time step flag.

= 0 Spatially constant time step.

= 1 Spatially variable time step. <default, highly recommended>

cfl — Courant number, typically 5.6 (see “Multistage Runge-Kutta Scheme” pp. 3.) If $ivtstp = 0$, *cfl* is the maximum Courant number, usually located somewhere near the leading edge at the blade surface. If $ivtstp = 1$, the Courant number will equal *cfl* everywhere. <5.>

ipc — Preconditioning flag, (see “Preconditioning” pp. 4.)

= 0 No preconditioning. <default>

= 1 Preconditioning using the Merkel, Choi, Turkel scheme. Should give a substantial speedup for low Mach numbers < 0.3.

= 2 Solves the equations using the preconditioning variable set, but sets the preconditioning matrix to the identity matrix. Used to debug the preconditioning routines.

pck — Constant limiter (Turkel's parameter *k*) for preconditioning. The denominator in the preconditioning matrix is limited to be > $pck \times q'^{2}_{\text{ref}}$, where q'_{ref} is a reference velocity set by *refm*. *Pck* is typically 0.1 - 0.3, but larger values may be necessary for stability. <0.15>

refm — Reference relative Mach number used to find q'_{ref} described above. Should be approximately the largest Mach number expected in the flow. If the code blows up, try increasing *refm* by 0.1. Convergence is mildly sensitive to *refm* and *pck*, so try to keep these values as small as possible. <*amle*>

***&nam3* - Boundary Condition & Code Control**

itmax — Number of iterations, typically 500-1000 per run, but 1000-3000 may be needed for a converged solution.

iresti — Read input restart file flag. Restart files are in PLOT3D format in the relative frame.

= 1 Read restart file from unit 2.

else No action taken. <default>

iresto — Write output restart file flag.

= 1 Write restart file to unit 3. <default>

else No action taken.

ibcin — Inlet boundary condition flag. In all cases P_0 and T_0 are fixed at the inlet. For subsonic flow a Riemann invariant is extrapolated from the interior to determine the meridional velocity *u*. *Ibcin* determines how v_θ is determined.

= 1 The θ-velocity component is held constant at the initial value based on *amle* and *alle*. <default>

= 2 Supersonic inflow - all quantities are fixed at the inlet.

= 3 The flow angle α is held constant at the initial value based on *amle* and *alle*. This option may not be well-posed for transonic compressors, where the inflow angle may be set by unique incidence.

ibcex — Exit boundary condition flag. Three conservation variables are extrapolated to the exit. *Ibcex* determines how the exit pressure *p* is determined.

= 1 The exit pressure is held constant at *prat*. Good for subsonic outflow. <default>

= 2 Supersonic outflow. *P* is extrapolated to the boundary.

= 3 The average exit pressure is held constant at $\bar{p} = prat$, but *p* may vary blade-to-blade. Based on characteristic boundary conditions developed by Giles. Especially good for transonic turbines with shocks leaving the trailing edge.

ires — Iteration increment for writing residuals in the output file. Typically 10. If the code is blowing up, set $ires = 1$ to print the size and location of the maximum residual at each iteration

icrnt — Iteration increment for updating the time step. Typically 50. If *icrnt* is very large, the residual history may be discontinuous where Δt is recalculated, especially at restarts. <50>

ixrm — Flag for additional output of blade coordinates and radius.

= 0 No output <default.>

= 1 x, y, and r are printed in the output file.

mioe — Mass flow output flag for residual history. For transonic fans the inflow may respond slowly to a change in back pressure, so the inlet mass flow can be monitored for convergence (*mioe* = 1). For turbines the inflow may choke quickly so the outflow can be monitored (*mioe* = 2). In general the mass flow error is a good measure of convergence and accuracy and should converge to a fraction of a percent (e. g., < 0.003, *mioe* = 3).

= 1 Inlet mass flow history is written.

= 2 Exit mass flow history is written.

= 3 Mass flow error $1 - \dot{m}_{out}/\dot{m}_{in}$ is written. <default>

***&nam4* - Flow Parameters**

Since the user may not know the inlet whirl or exit pressure at the boundaries of an arbitrary grid, it is assumed that the absolute Mach number *amle* and flow angle *alle* are known at the leading edge, and relative flow angle *bete* is known at the trailing edge. In other words, initial conditions are input in terms of the blade leading edge and trailing edge velocity triangles. An analytic solution of the 1-D continuity equation is used to determine the inlet conditions. At the inlet, either the θ-velocity component *v* or the flow angle α is held fixed at the initial value. For subsonic inflow the *m*-velocity component and mass flow change as the solution develops in response to the exit pressure ratio *prat*.

p0in — $P_{0\,in}/P_{0\,ref}$ at the inlet. Usually <1.0>, but can be varied to simulate a downstream blade row in a multistage environment.

t0in — $T_{0\,in}/T_{0\,ref}$ at the inlet. Usually <1.0>, but can be varied to simulate a downstream blade row in a multistage environment.

prat — Ratio of the hub exit static pressure to the reference total pressure, $prat = p_{hub\,exit}/P_{0\,ref}$.

amle — Nominal Mach number at the leading edge. Used for initial conditions. Note that the inlet Mach number will change with the solution for subsonic inflow.

alle — Absolute flow angle at the leading edge in degrees. <Default = 0. for axial inflow.>

bete — Relative flow angle at the trailing edge in degrees. Can be measured directly from a grid. Used for initial conditions only. <0.>

g — Ratio of specific heats. <$\gamma = 1.4$ for air>

***&nam5* - Viscous Parameters**

ilt — Inviscid, Laminar, or Turbulent analysis, with turbulence model options.

= 0 Inviscid Euler solution. The remaining viscous parameters are not used if *ilt*=0.

= 1 Laminar. Viscous terms are calculated, but the turbulent viscosity is zero.

= 2 Turbulent using the Baldwin-Lomax turbulence model (3). <default>

= 3 Turbulent using the Cebeci-Smith turbulence model (4). This model works well for turbine heat transfer but may overpredict losses for transonic compressors.

= 4 Fully turbulent using the baseline Wilcox k-ω turbulence model (6, 7).

= 5 Turbulent with transition using the low Reynolds number Wilcox k-ω turbulence model (6, 7).

jedge — The last j-index searched for the blade turbulent length scale. For the Baldwin-Lomax turbulence model $(ilt = 2)$, *jedge* should be a grid line slightly bigger than the largest expected blade boundary layer. For the Cebeci-Smith turbulence model $(ilt = 3)$, *jedge* should be a grid line slightly bigger than half the largest expected blade boundary layer. <10>

NOTE: *Jedge* is also used to turn off the artificial viscosity in viscous layers. The artificial viscosity is multiplied by a linear function of j-index that goes from 1 at j=*jedge* to 1/*jedge* at the wall. This improve viscous loss predictions. *Jedge* is set to 1 internally for inviscid flow.

renr Reynolds number per unit length based on reference conditions, $renr = \rho_0 c_0 / \mu_0$. Must have units of $[1/\text{grid units}]$. Generally much larger that a conventional "free-stream" Reynolds number. For example, for standard conditions and a grid with units of [ft],

$$\begin{aligned} \rho_0 &= 0.0765 \text{ lb}_m/\text{ft}^3 \\ c_0 &= 1116.7 \text{ ft/sec} \\ \mu_0 &= 3.99\times10^{-7} \text{ lb}_f\text{sec}/\text{ft}^2 = 1.285\times10^{-5} \text{ lb}_m/\text{ft sec} \\ renr &= 0.0765 \times 1116.7 / 1.285\times10^{-5} \\ &= 6.65\times10^{6}/\text{ft} \end{aligned}$$

prnr Prandtl number. <0.7 for air>

tw Normalized wall temperature, $tw = T_{\text{wall}}/T_{0\text{ ref}}$.

= 0 Adiabatic wall boundary conditions are used.

= 1 $T_{\text{wall}} = T_{0\text{ ref}}$.

else $T_{\text{wall}} = tw$.

vispwr Exponent for laminar viscosity power law. <default = 0.667 for air>

$$\mu/\mu_0 = (T/T_{0\text{ ref}})^{vispwr}$$

cmutm Value of $\mu_{\text{turb}}/\mu_{\text{lam}}$ at which transition is assumed to occur. Baldwin and Lomax recommend 14. Can be increased to move transition downstream or vice-versa. If *cmutm* = 0, the flow is fully turbulent. <14.>

hrough Surface roughness height in turbulent wall units h^+. Implemented in both the Baldwin-Lomax model (*ilt*=2) and the Cebeci-Smith model (*ilt*=3) using the Cebeci-Chang roughness model. $hrough \le 4$ gives a hydraulically-smooth surface.

tintens Freestream turbulence intensity as a decimal. Determines the inlet value of k for the k-ω model. Can be set very small, but a value of zero will give a laminar solution. <0.01>

tlength Turbulent length scale, $tlength = \sqrt{k/\omega}$ in the same units as the grid. *Tlength* is used to determine the inlet value of ω. *Tlength* usually has little effect on the solution, except for transition location. It may be necessary to vary *tlength* over several orders of magnitude to find the value that moves the transition point. Wilcox recommends a length scale of 0.01 to 0.001 of δ for predicting transition. For cascades this is the order of $1\times10^{-4} \times \text{pitch}$, or $0.1 \times \text{grid}$ spacing at the wall. Choosing *tlength* such that the Reynolds number based on *tlength* ~ 1 seems to give reasonable results, so the default for *tlength* is <1/*renr*>.

NOTE: RVCQ3D.300 used the freestream turbulent viscosity *tmuin* as the input parameter instead of *tlength*. A length scale seemed to be a more useful parameter in the 3-D SWIFT code, and RVCQ3D was changed for compatibility. The freestream turbulent viscosity is printed in the output for reference. Choosing *tlength* such that the freestream turbulent viscosity is about 0.1 usually gives good results.

***&nam6* - Blade Row Data**

omega Normalized blade row rotational speed, $\Omega/c_{0\text{ ref}}$ where Ω is the wheel speed in radians per second, and $c_{0\text{ ref}}$ is the reference speed of sound. *Omega* has dimensions of [1/grid units]. *Omega* is positive in the positive θ-direction, and tends to be negative for most GRC geometries. <0.>

nblade Number of blades, $nblade \geq 1$. <1>

If $nblade = 1$, the problem is treated as a linear cascade with radius = 1 and stream surface thickness = 1.

If $nmn > 1$, the problem is treated as an annular cascade with pitch $= 2\pi r/nblade$.

NOTE: Since grids are generated independently using the GRAPE code, it is possible that the pitch determined from the grid will not match the pitch determined from *nblade*. In that case RVCQ3D prints a warning in the output file, and rescales the θ-coordinates of the grid to give the correct pitch.

nmn Number of stream surface input points, $1 \leq nmn \leq 50$, <1>.

If $nmn = 1$, the problem is treated as a linear cascade with radius = 1 and stream surface thickness = 1. The input may be defaulted, but three blank lines **MUST** be included at the bottom of dataset as place holders for the stream surface input.

If $nmn > 1$ the stream surface input is read, as described below.

Stream Surface Input

The stream surface coordinates are input as m-coordinates *srsp*, radii *rmsp*, and stream surface thicknesses *besp* (fig. 5.) *Rmsp* and *besp* have first-order effects on the flow solution and should be picked carefully. For a crude model use the mid-span radius for *rmsp* and the blade height for *besp*. At NASA Glenn, solutions from Katsanis' axisymmetric throughflow code MERIDL that gives a detailed stream tube shape, or a blade design from Crouse's compressor design code on a conical section are often used. Information about endwall boundary layer blockage can be subtracted from *besp* if known.

The coordinates must be in the same units as the grid coordinates. They are spline fit to the grid, so they must be smooth and monotonically increasing. Points are extrapolated if *srsp* does not span the m-coordinates of the grid. The coordinates are read using an unformatted ascii read, e.g.:

```
read(5,*)(srsp(i),i=1,nmn)
read(5,*)(rmsp(i),i=1,nmn)
read(5,*)(besp(i),i=1,nmn)
```

srsp m-coordinates where the radius and stream-surface height are specified. **MUST** be monotonically increasing.

rmsp r-coordinates of the stream-surface at points *srsp*. The magnitude of *rmsp* enters directly into the flow equations in terms like $r \times d\theta$, and has a first-order effect on the flow solution.

besp Stream-surface thickness. The magnitude of *besp* is unimportant - only its m-derivative effects the solution. *Besp* does effect the printed mass flow so it is usually set to 1. at the inlet and scaled from there.

RVCQ3D Output

Printed output from RVCQ3D is written to Fortran unit 6 (standard output.) The output is divided into several sections. The sections are commonly separated using an editor and plotted using any x-y plotting package that can read ascii column data. The output includes the following information:

1. The input variables are echoed back for reference, and any comments or warnings regarding the input are given.

2. A table of θ – averaged flow variables is given at the inlet and exit. Both absolute and relative quantities are given. These variables are either based on the initial guess or on a restart file, depending on how the code is started.

3. A convergence history gives maximum and RMS residuals of density, mass flow error, and exit flow properties $P_{0\,\mathrm{exit}}$ and $T_{0\,\mathrm{exit}}$ versus iteration.

4. A table of θ – averaged flow variables at the inlet and exit is repeated for the new solution. An approximate energy-averaging scheme is used. It gives a local representation of the average flow, not a mixed-out average.

5. Blade surface distributions of various quantities are given versus *m*. Values of y^+ for the first grid point are given to check the grid spacing for the turbulence model. y^+ should always be < 10, and should be < 2–3 for good prediction of losses. Maximum values of μ_T are given to identify transition points. With the Baldwin-Lomax and Cebeci-Smith models transition occurs where μ_T jumps abruptly from 0 to $\mu_T > cmutm$. With the k-ω model transition occurs more gradually but should be obvious as a rapid growth of μ_T.

Appendix - File Descriptions

Grid *XYZ*-File

Grids are stored using standard PLOT3D xyz-file structure. Grids can be read with the following Fortran code:

```
c     read grid coordinates
      read(1)m,n
      read(1)((x(i,j),i=1,m),j=1,n)),
     &       ((y(i,j),i=1,m),j=1,n))
```

Solution Q-File

Solution files are stored in standard PLOT3D q-file structure. Solution files can be read with the following Fortran code:

```
c     read q-file
      read(2)m,n
      read(2)emir,abir,renr,time
      read(2)(((qq(k,i,j),i=1,m),j=1,n),k=1,4)

c     additional geometry data and residual history
      read(2)mtl,mil,ires,nres,nmn,g,omega,yscl
      read(2)((resd(nr,l),nr=1,nres),l=1,5

c     further data for Dan Tweedt's post-processing codes
      read(2)((qq(k,i,n),i=1,m),k=1,4)
      read(2)cfl,dtmin,idum,idum
      read(2)(srsp(l),l=1,nmn)
      read(2)(rmsp(l),l=1,nmn)
      read(2)(besp(l),l=1,nmn)
```

The q-variables are given in the rotating frame of reference:

$$q = [\rho, \rho u, \rho v', e]$$

$$e = \rho\left(C_v T + \frac{1}{2}\rho(u^2 + v'^2)\right)$$

Solution k-ω File

The k-ω files are also stored in standard PLOT3D q-file structure for convenience. Solution files can be read with the following Fortran code

```
      read(7)m,n
      read(7)emir,abir,renr,float(iter)
      read(7)((wk(1,i,j),i=1,m),j=1,n),
     &       ((wk(2,i,j),i=1,m),j=1,n),
     &       ((tmu( i,j),i=1,m),j=1,n),
     &       ((re(  i,j),i=1,m),j=1,n)
```

where

$$\text{wk} = [k, \omega, \mu_{\text{turb}}, Re_{\text{turb}}]$$

References

1. Chima, R. V., “Explicit Multigrid Algorithm for Quasi-Three-Dimensional Viscous Flows in Turbomachinery,” J. Propulsion and Power, Vol. 3, no. 5, Sept. – Oct. 1987, pp. 397–405.

2. Chima, R. V., “Viscous Three-Dimensional Calculations of Transonic Fan Performance,” in *CFD Techniques for Propulsion Applications*, AGARD Conference Proceedings, No. CP-510, AGARD, Neuilly-Sur-Seine, France, Feb. 1992, pp. 21-1 to 21-19. Also NASA TM-103800.

3. Baldwin, B. S., and Lomax, H., “Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows,” AIAA Paper 78-257, Jan. 1978.

4. Chima, R. V., Giel, P. W., and Boyle, R. J., “An Algebraic Turbulence Model for Three-Dimensional Viscous Flows,” in *Engineering Turbulence Modeling and Experiments 2*, Rodi, W. and Martelli, F. editors, Elsevier pub. N. Y., 1993, pp. 775-784. Also NASA TM-105931.

5. Cebeci, T. and Chang, K. C., “Calculation of Incompressible Rough-Wall Boundary Layer Flows,” AIAA Journal, Vol. 16, July, 1978, pp. 730-735.

6. Wilcox, D. C., *Turbulence Modeling for CFD*, DCW Industries, Inc. La Canada, CA, 1994.

7. Chima, R. V., “A k-ω Turbulence Model for Quasi-Three-Dimensional Turbomachinery Flows,” NASA TM-107051, Jan. 1996.

8. Turkel, E., “A Review of Preconditioning Methods for Fluid Dynamics,” *Applied Numerical Mathematics*, Vol. 12, 1993, pp. 257-284.

9. Tweedt, D. L., Chima, R. V., and Turkel, E., “Preconditioning for Numerical Simulation of Low Mach Number Three-Dimensional Viscous Turbomachinery Flows,” AIAA Paper 97-1828, June 1997. Also NASA TM-113120.

10. Sorenson, R. L., “A Computer Program to Generate Two- Dimensional Grids About Airfoils and Other Shapes by Use of Poisson’s Equation,” NASA TM-81198, 1980.

11. Jameson, A., Schmidt, W., and Turkel, E., “Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes,” AIAA Paper 81-1259, June 1981.

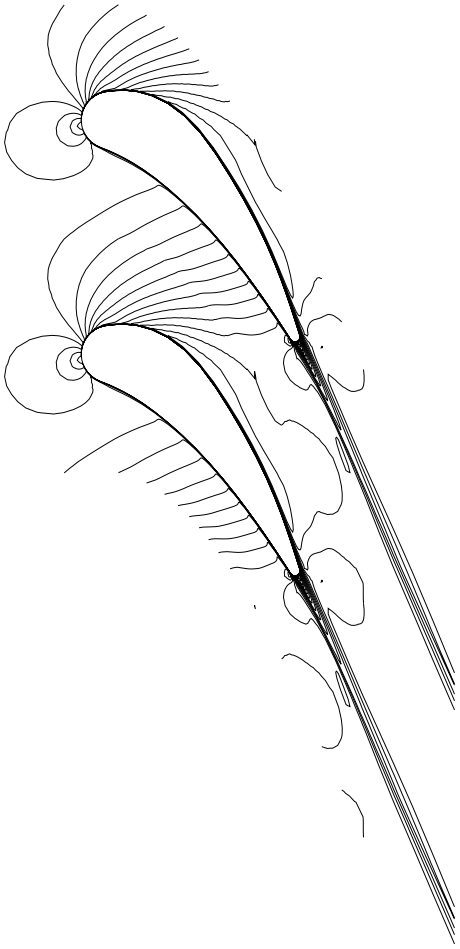

Figure 1 – Mach contours for a turbine vane.

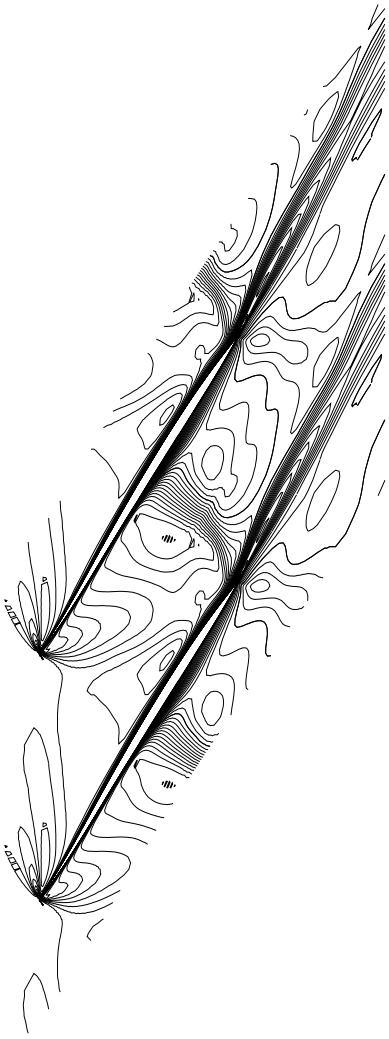

Figure 2 – Mach contours for a compressor rotor.

Figure 3 – Mach contours for a centrifugal impeller rotor.

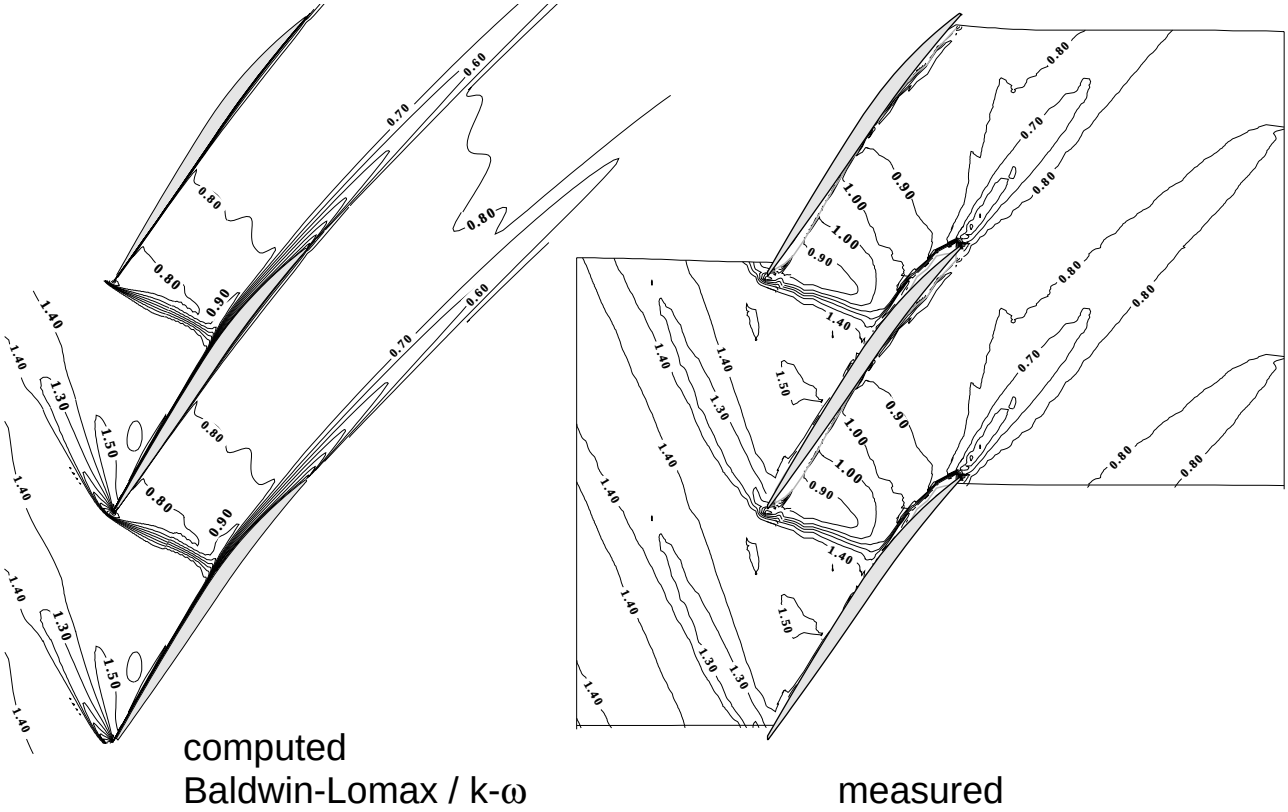

Figure 4 — Computed and measured contours of relative Mach number for a transonic compressor rotor.

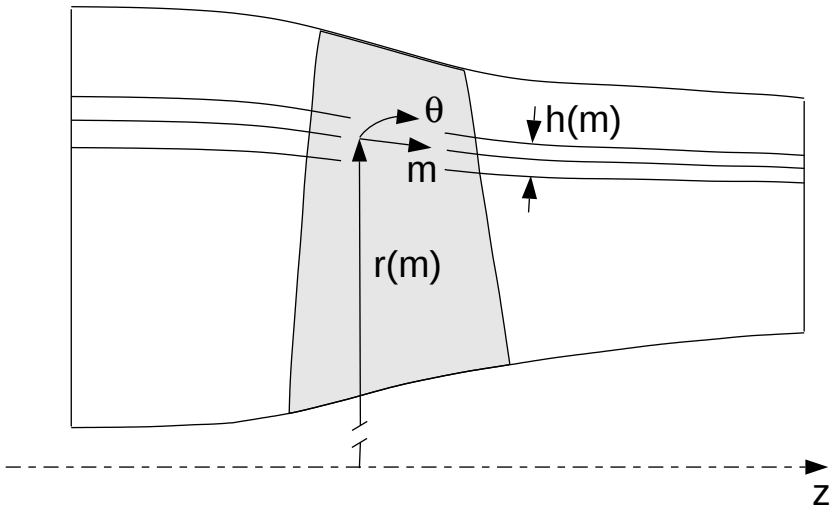

Figure 5 — Quasi-three-dimensional stream surface for a transonic compressor rotor.

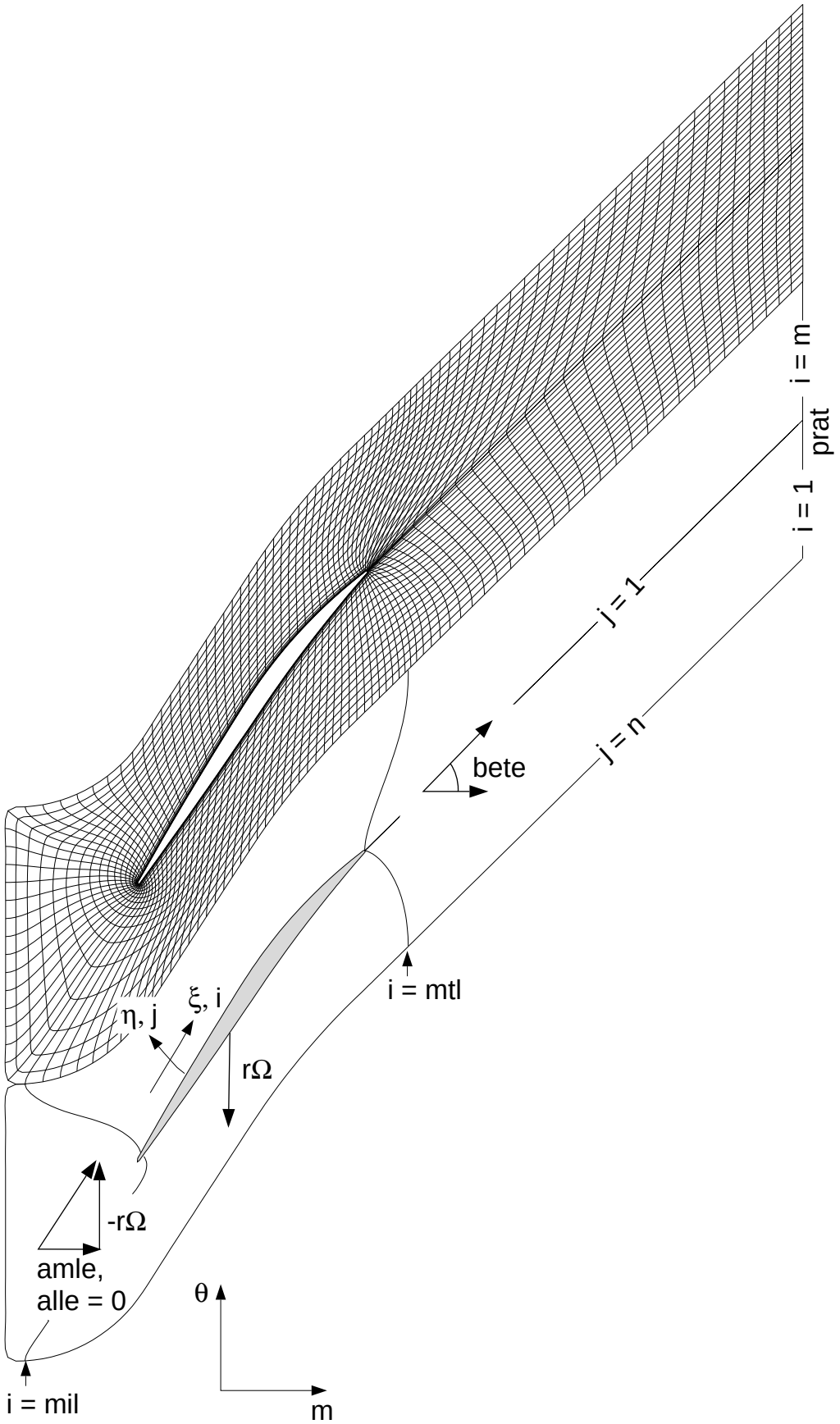

Figure 6 — Top: Computational grid for a transonic compressor rotor.
Bottom: Body-fitted coordinate system and index convention.